The Open Motion Planning Library 2.0

Weihang Guo[1], Theodoros Tyrovouzis[1], Emiliano Flores[1], Clayton W. Ramsey[1], Zachary K. Kingston[2*], Ioan A. Şucan[3*], Mark Moll[4], Lydia E. Kavraki[1,5]

***Abstract*—The Open Motion Planning Library (OMPL), first released in 2008, has become a cornerstone of the motion planning community, providing implementations of a wide range of state-of-the-art sampling-based algorithms. Over almost two decades of continuous development, we have steadily expanded the library with new planners, state spaces, and problem formulations. These additions range from asymptotically optimal and lazy planners to constrained motion planning and planning with temporal-logic goals. Building on this foundation, we introduce OMPL 2.0, a major evolution of the library that targets real-time motion planning through hardware acceleration and integrates seamlessly with modern AI research workflows. We also reflect on how OMPL and the field of motion planning have grown together over the years, and discuss the library's broader impact on the research community.**

I. Introduction

The history of motion planning can be traced back to the late 1970s, when the concept of configuration space [1] was introduced to formally describe robot motion planning problems. Early approaches focused on exact algorithms [2] and potential field methods [3, 4]. However, these techniques often struggled with the high computational complexity of realistic robotic systems, as research has proven that motion planning in configuration space is NP-hard [2]. In the 1990s, sampling-based motion planning emerged as a practical approach. Algorithms such as the probabilistic roadmap (PRM) [5], expansive-space tree (EST) [6], and rapidly-exploring random trees (RRT) [7] developed before 2000, approximate the connectivity of the configuration space through random sampling and enable efficient planning in high-dimensional spaces. These approaches quickly became a dominant paradigm in robotics. Several early sampling-based planners were accompanied by theoretical analyses that characterized their trade-offs [8, 9]. Subsequent research introduced improved sampling strategies and algorithmic refinements. Later, in the 2010s, asymptotically optimal planners such as PRM*, RRT* [10], and SST [11] provided convergence guarantees to optimal solutions. As sampling-based motion planning algorithms rapidly developed, implementing and fairly comparing new planners became increasingly challenging. Researchers often needed to reimplement existing algorithms and supporting data structures, making reproducibility and benchmarking difficult.

To address the reproducibility and benchmarking challenges, the Open Motion Planning Library (OMPL) [12] was developed and released in 2008 as an open-source software framework for motion planning. OMPL provides implementations of a wide range of sampling-based planners together with reusable and flexible abstractions for defining planning problems, enabling researchers and practitioners to easily evaluate algorithms, benchmark with other planners, and integrate motion planning into robotics applications.

Almost two decades have passed since OMPL's first release. Over this period, motion planning research, computer architectures, and the broader robotics ecosystem have all advanced substantially. OMPL has evolved alongside these changes, continually incorporating new planners, state space representations, and problem formulations. In this paper, we introduce OMPL 2.0, a major evolution of the library that targets real-time motion planning through hardware acceleration and integrates with modern AI research workflows.

II. The Open Motion Planning Library

Before introducing new features of OMPL 2.0, we recap the main components of OMPL. This design, introduced in the original OMPL paper [12], remains largely unchanged and has stood the test of time. OMPL is structured as a set of modular components that correspond closely to the fundamental concepts of sampling-based motion planning. The core abstractions include state space, state validator, state sampler, start and goal, and motion planners, which together define the planning problem.

The *state space* defines the representation and topology of the robot's configuration space, along with its limits and any kinematic or dynamic constraints. A *state validator* provides a generic interface for collision checking, allowing users to connect any collision checking library or simulator. Finally, a *state sampler* draws configurations uniformly or from a Gaussian distribution over the state space. With these foundational components in place, users can select or build their own *motion planner* and specify *start* and *goal* states to solve the problem. Beyond solving individual planning problems, the *benchmark* component provides extensible infrastructure for running evaluations and analysis across multiple planners [13]. Results and metadata are stored in a database to support reproducibility.

A key design choice in OMPL is to focus exclusively on motion planning algorithms and to not include specific representations of robots, workspaces, or collision detection.

[1]WG, TT, EF, CWR, LEK are with the Department of Computer Science, Rice University, Houston TX, USA. {wg25,tt88,ef55,cwr3,kavraki}@rice.edu

[2]ZKK is with the Department of Computer Science, Purdue University, West Lafayette, IN. zkingston@purdue.edu

[3]IAŞ is with Waymo, LLC, Mountain View, CA

[4]MM is with Metron, Inc., Reston, VA

[5]LEK is also affiliated with the Ken Kennedy Institute at Rice University

*Work by these authors was done while they were at Rice University.

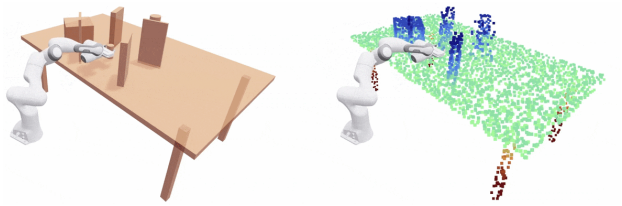

Fig. 1: OMPL 2.0 supports planning with realistic robots and environments represented by meshes or point clouds.

Instead, OMPL relies on external software components to provide these capabilities. This minimalist design allows the library to remain general and easily integrated with a wide range of robotics systems and simulation environments. OMPL is a community effort, with contributions from many researchers over the years.

III. Two Decades of Adoption

OMPL has become a widely used tool in the motion planning community. The original paper has accumulated over 2,400 citations, reflecting sustained and widespread adoption. By providing carefully maintained reference implementations of canonical planners, it has helped standardize how the community benchmarks and communicates algorithmic contributions, allowing researchers to focus on novel contributions rather than reimplementing baselines.

A key factor behind this adoption is OMPL's separation from robot models, workspace representations, and collision geometry. By defining a clean interface around state spaces, validity checkers, and planners, OMPL integrates naturally with external systems without imposing constraints on how they are built. OMPL can be called from MoveIt [14], robot simulators such as CoppeliaSim [15] and Flightmare [16], industrial frameworks like Tesseract [17], and research toolkits including AIKIDO [18], EXOTica [19], and the Kautham Project [20]. OMPL's planners have also been used in task-and-motion planning [21] and mobile robot navigation via Nav2 [22], while tools such as HyperPlan [23], Robowflex [24], and MotionBenchMaker [25] build on its benchmarking infrastructure for standardized algorithm evaluation. OMPL has also been used in non-robotics applications, such as modeling the conformational flexibility of large macromolecules [26].

IV. OMPL 2.0: Key Technical Improvements

Rather than emerging from a single landmark release, OMPL has matured through more than a decade of incremental development. Since the publication of the original paper, over 30 versions have been released along the path from OMPL 0.x through OMPL 1.0 to OMPL 2.0, each contributing new algorithms and capabilities. These updates reflect both advances in motion planning research and the evolving needs of the robotics community.

a) Problem Set Expansion: OMPL 2.0 now supports a wider range of planning paradigms, including asymptotically optimal planners [27, 28] and lazy planning methods [29]. Beyond extending its set of algorithms, OMPL 2.0 has also broadened the kinds of planning problems it can express. Constrained motion planning [30] enables any sampling-based planner to operate directly on an implicit constraint manifold, supporting practical problems involving contact constraints and closed kinematic chains. For task-level specifications, LTLPlanner [4] is a kinodynamic planner that produces trajectories satisfying a given linear temporal logic formula by searching the cross product of the continuous state space and the discrete space of the formula's accepting traces. In addition, the library has expanded its set of state space representations to cover a broader class of problems, including discrete state spaces, time-augmented spaces for planning under temporal constraints, and non-Euclidean spaces such as Dubins and Reeds–Shepp for car-like vehicles. Together, these additions significantly broaden the types of planning problems that can be addressed within OMPL 2.0. The ever-growing number of planning algorithms, each with its own parameters, can make it challenging to select an appropriate planner for a specific robot or environment. We have shown that hyperparameter tuning can be used as an effective tool for planning algorithm selection and tuning [23].

b) Performance Evolution: Beyond advances in planning algorithms, OMPL 2.0 integrates VAMP [31], which leverages CPU SIMD instructions to perform collision checking and forward kinematics with fine-grained, hardware-efficient parallelism. With this integration, planners can find solutions in microseconds, and aggregate solution rates can reach the kilohertz range, all without specialized hardware such as a GPU. OMPL now also includes CAPT [32], which extends the SIMD philosophy to point cloud collision checking. These capabilities open up new opportunities for applications that require extremely fast planning in dynamic environments, including reactive planning and task-and-motion planning. The OMPL 2.0's integration with VAMP and CAPT also makes it straightforward to apply motion planning to realistic robots operating in real-world scenes (see Fig. 1).

c) Modernized Development Infrastructure: When OMPL was first released, many of the software development tools and practices now common in open-source projects were not yet widely adopted. At the time, OMPL used Py++ [33] to generate its Python bindings. The tool is no longer maintained. To address this, OMPL 2.0 has transitioned to nanobind [34], a modern C++ binding framework that provides a lightweight and efficient interface between C++ and Python while simplifying the maintenance of binding code [35]. In addition, OMPL 2.0 has adopted contemporary development infrastructure such as GitHub Actions for continuous integration, enabling automated building and testing across multiple platforms. The project also employs automated pipelines to build and publish Python wheels to PyPI, significantly simplifying installation and distribution.

d) Streamlining: Earlier versions of OMPL included OMPL.app, a graphical front end that demonstrated integration with external libraries for mesh loading, collision checking, and visualization. Over the past decade, the robotics ecosystem has matured, with widely used simulation platforms and collision detection libraries that integrate naturally with OMPL, making a dedicated graphical interface no longer necessary.

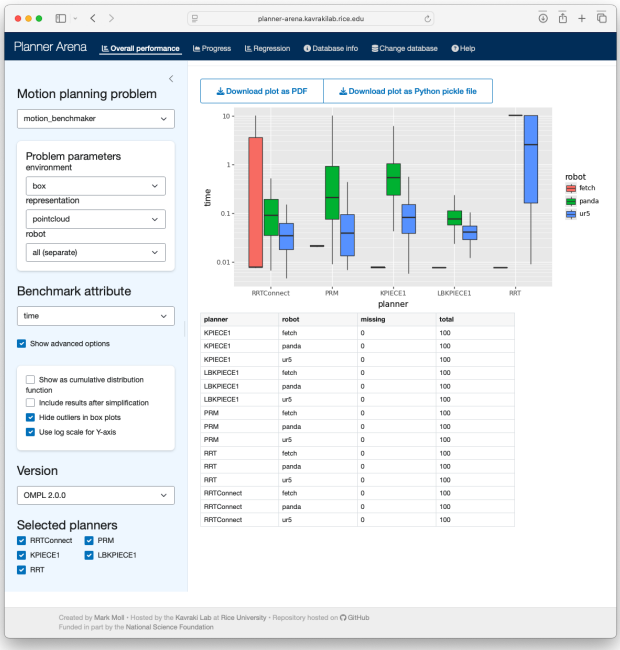

Fig. 2: Planner Arena allows users for benchmarking motion planning algorithms.

OMPL.app has therefore been removed in OMPL 2.0, allowing the library to focus on efficient planning infrastructure and tighter integration with external robotics software.

V. Project Community and Organization

The goal from the beginning of the OMPL project has always been to create a resource that is useful for research, education, and real-world use. This goal has been achieved, but it is also something that requires ongoing effort to retain relevance. Long-term software maintenance and support remain a challenge in an academic environment. Shortly after the initial release, we organized a few tutorials at robotics conferences to get researchers started with OMPL and help start a user community. Improved documentation and more general awareness of OMPL have reduced the need for this. We provide the link to the most recent OMPL tutorial.[1]

We also decided early on to spend significant time on building an extensible infrastructure for benchmarking motion planning algorithms [13]. This has been invaluable in quantifying how new planners advance the state of the art. More recently, we have also generated large datasets of realistic hard motion planning problems (and made the tooling for generating such datasets available as well) [25]. Benchmark results can be visualized through an interactive browser-based tool called Planner Arena.[2] Fig. 2 shows sample results for a parameterized benchmark. The parameters in this case are (1) the specific motion planning problem, (2) how the environment is represented (mesh vs. point cloud), and (3) the type of robot (a Fetch, a Panda, and a UR5 in this example). The UI enables comparison of planning algorithms on all robots for a single problem, on a single robot on all problems, etc. In each case, there are many performance metrics that can be visualized, such as planning time and solution path length. The table in the bottom right shows the number of missing values: if a planner was not able to find a path on some runs, the path length would be considered missing for those runs. In summary, Planner Arena encourages exploration and a nuanced characterization of planner performance.

The abstractions that OMPL imposes on planning algorithms are minimal and make it straightforward to develop prototypes of planners. Evaluation in simplified environments (e.g., a point robot in 2D) makes testing easy, before deployment in realistic environments without having to change the planning algorithm implementation itself.

Many motion planning researchers have developed new motion planning algorithms using OMPL. Oftentimes, the original authors contributed their implementations to the OMPL repository. For example, Karaman and Frazzoli contributed the initial RRT* implementation to OMPL shortly after publication [10]. Bekris et al. contributed implementations of their asymptotically near-optimal algorithms [11]. Gammell et al. have contributed many algorithms such as BIT*, ABIT*, AIT*, EIT*, and AORRTC [27, 36]. Orthey et al. have contributed implementations of their work on multi-level planning [37] and planning in space-time [38]. This is by no means an exclusive list.[3]

Over time, the OMPL community has grown to include many AI researchers, who often use OMPL simply as a black box. Because Python is the primary language for integrating software components in the AI community, OMPL's Python bindings have become increasingly important. Originally, these bindings were intended mainly to lower the barrier to entry for beginning programmers. Today, their performance penalty is small enough to be acceptable for many use cases, though we still recommend the C++ API directly when optimal performance is needed.

VI. The Next Decade of OMPL

Looking ahead, we see several directions that will shape the future development of OMPL. Motion planning is increasingly used as a component within larger decision-making systems: task and motion planning [21] combines symbolic reasoning with geometric planning, planning under uncertainty, such as POMDPs [39], demands rapid replanning as beliefs evolve, and reactive planning [40] requires algorithms that quickly adapt to changing environments. Supporting these paradigms will require continued investment in fast, anytime planning and interfaces that facilitate integration with higher-level reasoning. Beyond robotics pipelines, OMPL can also serve the broader AI ecosystem. By providing a Model Context Protocol server and agent skills, we aim to let LLM-based agents invoke motion planning directly as a tool.

As robotic systems scale from single manipulators to teams of cooperating robots, multi-robot planning in composite

[1] https://kavrakilab.org/icra-2026-ompl-tutorial/
[2] https://planner-arena.kavrakilab.rice.edu/
[3] A list of contributors is maintained in https://ompl.kavrakilab.org/developers.html

configuration spaces becomes a central challenge. Recent work has shown that sampling-based methods can be extended to these settings by exploiting problem structure [41], and future versions of OMPL aim to provide support for multi-robot coordination. At the same time, GPU-parallel approaches to tree construction and collision checking [42] offer the potential for massive speedups. Finally, emerging applications in soft and growing robots [43] involve high-dimensional, continuously deformable state spaces that challenge existing representations and will require extending OMPL's state space abstractions to new physical domains.

These directions share a common theme: the core abstractions of sampling-based planning remain the right foundation, but must be extended to meet the demands of increasingly complex robotic systems. We see OMPL 2.0 not as an endpoint, but as a starting point for the next decade of motion planning research.

Acknowledgements

The authors gratefully acknowledge the many contributors whose work over the years has shaped the OMPL 2.0. All contributors are acknowledged on the OMPL web page. Contributions can be found in the OMPL GitHub repository. We note, however, that commit counts and lines of code are imperfect indicators of impact. Several significant contributions, such as the initial RRT* implementation, predate the move to GitHub and do not appear in those statistics. The authors would like to thank current and former members of the Kavraki Lab, as well as the broader motion planning community, whose discussions, feedback, and support have been instrumental to this work. This work was supported in part by NSF OAC-2411219.

References

[1] T. Lozano-Perez. "Spatial planning: A configuration space approach". In: *IEEE Transactions on Computers* C-32.2 (1983), pp. 108–120.

[2] J. Canny. *The complexity of robot motion planning*. MIT press, 1988.

[3] J. Barraquand, B. Langlois, and J.-C. Latombe. "Numerical potential field techniques for robot path planning". In: *IEEE Transactions on Systems, Man, and Cybernetics* 22.2 (1992), pp. 224–241.

[4] A. Bhatia, M. R. Maly, L. E. Kavraki, and M. Y. Vardi. "Motion planning with complex goals". In: *IEEE Robotics & Automation Magazine* 18.3 (2011), pp. 55–64.

[5] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[6] D. Hsu, J.-C. Latombe, and R. Motwani. "Path planning in expansive configuration spaces". In: *International Journal of Computational Geometry & Applications* 09.04n05 (1999), pp. 495–512.

[7] J. Kuffner and S. LaValle. "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation*. Vol. 2. 2000, 995–1001 vol.2.

[8] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe. "Analysis of probabilistic roadmaps for path planning". In: *IEEE Transactions on Robotics and Automation* 14.1 (1998), pp. 166–171.

[9] L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. "Randomized query processing in robot path planning". In: *Proceedings of the twenty-seventh Annual ACM Symposium on Theory of Computing*. 1995, pp. 353–362.

[10] S. Karaman and E. Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894.

[11] Y. Li, Z. Littlefield, and K. E. Bekris. "Asymptotically Optimal Sampling-Based Kinodynamic Planning". In: *International Journal of Robotics Research* 35 (Apr. 2016), pp. 528–564.

[12] I. A. Şucan, M. Moll, and L. E. Kavraki. "The Open Motion Planning Library". In: *IEEE Robotics and Automation Magazine* 19.4 (Dec. 2012). https://ompl.kavrakilab.org, pp. 72–82.

[13] M. Moll, I. A. Şucan, and L. E. Kavraki. "Benchmarking Motion Planning Algorithms: An Extensible Infrastructure for Analysis and Visualization". In: *IEEE Robotics & Automation Magazine (Special Issue on Replicable and Measurable Robotics Research)* 22.3 (Sept. 2015), pp. 96–102.

[14] S. Chitta, I. Şucan, and S. Cousins. "MoveIt! [ROS Topics]". In: *IEEE Robotics and Automation Magazine* 19.1 (2012), pp. 18–19.

[15] E. Rohmer, S. P. Singh, and M. Freese. "V-REP: A versatile and scalable robot simulation framework". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 1321–1326.

[16] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza. "Flightmare: A Flexible Quadrotor Simulator". In: *Conference on Robot Learning*. 2020.

[17] L. Armstrong. *Tesseract: Motion Planning Environment*. https://github.com/tesseract-robotics/tesseract. 2018.

[18] M. Koval, J. King, J. Lee, G. Lee, B. Hou, P. Velagapudi, C. Liddick, D. Yi, A. V. Mandalika, S. Niyaz, E. Gordon, et al. *AIKIDO: Artificial Intelligence for Kinematics, Dynamics, and Optimization*. Version 0.4.0. 2020.

[19] V. Ivan, Y. Yang, W. Merkt, M. P. Camilleri, and S. Vijayakumar. "EXOTica: An Extensible Optimization Toolset for Prototyping and Benchmarking Motion Planning and Control". In: *Robot Operating System (ROS): The Complete Reference (Volume 3)*. Ed. by A. Koubaa. Cham: Springer International Publishing, 2019, pp. 211–240.

[20] J. Rosell, A. Pérez, A. Aliakbar, Muhayyuddin, L. Palomo, and N. García. "The Kautham Project: A Teaching and Research Tool for Robot Motion Planning". In: *IEEE International Conference on Emerging Technologies and Factory Automation*. 2014, pp. 1–8.

[21] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki. "An incremental constraint-based framework for task and motion planning". In: *The International Journal of Robotics Research* 37.10 (2018), pp. 1134–1151.

[22] S. Macenski, F. Martín, R. White, and J. G. Clavero. "The marathon 2: A navigation system". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2020, pp. 2718–2725.

[23] M. Moll, C. Chamzas, Z. Kingston, and L. E. Kavraki. "HyperPlan: A framework for motion planning algorithm selection and parameter optimization". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2021, pp. 2511–2518.

[24] Z. Kingston and L. E. Kavraki. "Robowflex: Robot motion planning with MoveIt made easy". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2022, pp. 3108–3114.

[25] C. Chamzas, C. Quintero-Peña, Z. Kingston, A. Orthey, D. Rakita, M. Gleicher, M. Toussaint, and L. E. Kavraki. "MotionBenchMaker: A Tool to Generate and Benchmark Motion Planning Datasets". In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022), pp. 882–889.

[26] B. Gipson, M. Moll, and L. E. Kavraki. "SIMS: A hybrid method for rapid conformational analysis". In: *PLOS ONE* 8.7 (2013). PMCID: PMC3720858, PMID: 23935893, p. 68826.

[27] T. S. Wilson, W. Thomason, Z. Kingston, and J. D. Gammell. "AORRTC: Almost-Surely Asymptotically Optimal Planning With RRT-Connect". In: *IEEE Robotics and Automation Letters* 10.12 (2025), pp. 13375–13382.

[28] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa. "Batch informed trees (bit*): Informed asymptotically optimal anytime search". In: *The International Journal of Robotics Research* 39.5 (2020), pp. 543–567.

[29] R. Bohlin and L. Kavraki. "Path planning using lazy PRM". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation*. Vol. 1. 2000, 521–528 vol.1.

[30] Z. Kingston, M. Moll, and L. E. Kavraki. "Exploring Implicit Spaces for Constrained Sampling-Based Planning". In: *International Journal of Robotics Research* 38.10–11 (Sept. 2019), pp. 1151–1178.

[31] W. Thomason, Z. Kingston, and L. E. Kavraki. "Motions in Microseconds via Vectorized Sampling-Based Planning". In: *IEEE International Conference on Robotics and Automation*. 2024, pp. 8749–8756.

[32] C. W. Ramsey, Z. Kingston, W. Thomason, and L. E. Kavraki. "Collision-Affording Point Trees: SIMD-Amenable Nearest Neighbors for Fast Collision Checking". In: *Robotics: Science and Systems*. 2024.

[33] R. Yakovenko. *Py++ - Boost.Python code generator*. https://github.com/ompl/pyplusplus. 2004.

[34] W. Jakob. *nanobind: tiny and efficient C++/Python bindings*. https://github.com/wjakob/nanobind. 2022.

[35] W. Guo, T. Tyrovouzis, and L. E. Kavraki. “Python Bindings for a Large C++ Robotics Library: The Case of OMPL”. In: *IEEE International Conference on Robotics and Automation*. 2026.

[36] M. P. Strub and J. D. Gammell. “Adaptively Informed Trees (AIT*) and Effort Informed Trees (EIT*): Asymmetric bidirectional sampling-based path planning”. In: *The International Journal of Robotics Research* 41.4 (2022), pp. 390–417.

[37] A. Orthey, S. Akbar, and M. Toussaint. “Multilevel motion planning: A fiber bundle formulation”. In: *The International Journal of Robotics Research* 43.1 (2024), pp. 3–33.

[38] F. Grothe, V. N. Hartmann, A. Orthey, and M. Toussaint. “ST-RRT*: Asymptotically-Optimal Bidirectional Motion Planning through Space-Time”. In: *IEEE International Conference on Robotics and Automation*. 2022, pp. 3314–3320.

[39] Y. Liang, E. Kim, W. Thomason, Z. Kingston, L. E. Kavraki, and H. Kurniawati. “Think Fast and Far: Long-Horizon Online POMDP Planning via Rapid State Sampling”. In: *International Journal of Robotics Research* (2026).

[40] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots. “STORM: An integrated framework for fast joint-space model-predictive control for reactive manipulation”. In: *Conference on Robot Learning*. PMLR. 2022, pp. 750–759.

[41] W. Guo, Z. Kingston, K. Hang, and L. E. Kavraki. “Efficient Multi-Robot Motion Planning for Manifold-Constrained Manipulators by Randomized Scheduling and Informed Path Generation”. In: *IEEE Robotics and Automation Letters* 11.4 (2026), pp. 4385–4392.

[42] C. H. Huang, P. Jadhav, B. Plancher, and Z. Kingston. “pRRTC: GPU-parallel RRT-Connect for fast, consistent, and low-cost motion planning”. In: *IEEE International Conference on Robotics and Automation*. 2026.

[43] Y. Gao, L. Chen, P. Bhovad, S. Wang, Z. Kingston, and L. H. Blumenschein. “Parallel Simulation of Contact and Actuation for Soft Growing Robots”. In: *Soft Robotics* (2026).